CAR-TR-470
CS-TR-2341

N00014-89-J-1854
November 1989

IMAGE SMOOTHING AND DIFFERENTIATION WITH MINIMAL-CURVATURE FILTERS

Isaac Weiss

Center for Automation Research
University of Maryland
College Park, MD 20742-3411

COMPUTER VISION LABORATORY

CENTER FOR AUTOMATION RESEARCH

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

4

CAR-TR-470
CS-TR-2341

N00014-89-J-1854
November 1989

IMAGE SMOOTHING AND DIFFERENTIATION WITH MINIMAL-CURVATURE FILTERS

Isaac Weiss

Center for Automation Research
University of Maryland
College Park, MD 20742-3411

Abstract

The fundamental problem of smoothing and differentiating of noisy images has been previously approached in two different ways: 1) Minimization of a smoothness functional, a theoretically well understood procedure but one that involves the solution of a very large system of equations involving all the pixels of the image, for each image. 2) Use of small scale, ready made filters for local smoothing. The process is computationally cheap but generally *ad hoc* and not very reliable. This paper offers a way to combine the advantages of the two approaches. We construct general filters for local windows of the image, derived from maximization of smoothness or "regularization" theory. In this way the theoretically robust minimization process becomes suitable for practical implementation, possibly in real time, and is readily adaptable to local image properties. Filters for more reliable derivatives are also being derived.

Keywords: smoothing, surface fitting, differentiation, interpolation, minimal curvature, filters, splines, regularization, optimal shape

The support of the Office of Naval Research under Grant N00014-89-J-1854 is gratefully acknowledged.

1. Introduction

Smoothing of noisy images has preoccupied research in vision and image processing ever since its beginning, and a variety of methods have been developed that partly overcome the problem but leave much to be desired. Strictly speaking the problem is underdetermined since there is no way to distinguish with certainty between data and noise, and to interpolate between data points, so some reasonable assumption must be made. Of course a theory with weak extraneous assumptions is preferable, but one often finds it unavoidable to make rather strong simplifying assumptions in order to restrict the solution space and improve computational efficiency. Thus with current methods one has to trade off generality for efficiency. In this paper we show that one can relax the need for this trade-off to a large extent and achieve a reasonable degree of both generality and efficiency.

We first briefly review two approaches that represent different choices in this trade-off and then present a theory that combines them.

The *minimum assumptions* approach consists of a minimization of some cost (or energy) functional E, which is a function of the unknown smoothed image f and the data g. (f, g may be grey levels, depths, or any interesting function defined at image coordinates $\mathbf{x}$.) The functional can be expressed generally as

$$E = \int V(f, g) + \int S(f)$$

The first term is a metric V that measures the distance between the desired (minimizing) shape f and the data g; the smaller the better. The second term integrates a smoothing function $S(f)$, which contains derivatives or curvatures of the shape f. This generalizes the idea that Grimson [1981] called informally "no news is good news", i.e. the surface's departure from smoothness is minimal, subject to the requirement of small distance from the data. The exact forms of V, S depend on the theory. The energy minimization principle was used by Horn [1983] for curves and by Barrow and Tenenbaum [1981] for interpreting line drawings. Poggio *et al.* [1987] have put this principle in the more rigorous mathematical framework of the "regularization theory" of Tikhonov and Arsenin. This theory

shows that indeed, under weak and general assumptions, the smoothing term $\int S$ turns an ill-posed problem into a well-posed one in the sense of Hadamard ([1923]), i.e. an underdetermined problem turns into one that has a unique and stable solution. Blake and Zimmerman [1987] studied the convergence and other properties of various cost functions, Weiss [1986] investigated cost functions that adapt themselves to local properties of the shape, and Terzopoulos [1988] studied implementation methods based on this general framework.

In implementing the method, each pixel is assigned an unknown variable (or several unknowns) such as the depth of the image at that point; the cost function is calculated in terms of the unknowns. Minimizing the cost function leads to equations for these unknowns. In this way one obtains a large set of coupled equations involving all the pixels. Solving the system for 3-D shapes has shown the validity of the basic approach but also the prohibitive cost, even with acceleration methods such as multiple grids. Furthermore, since the solution has to be done for the whole image at once, numerical problems arising from a small area can impede the whole solution. Obviously one would prefer a more local solution, that involves fewer pixels at a time and is not sensitive to every numerical ill-conditioning arising from anywhere in the image.

The other approach to smoothing uses filters that are calculated in advance for a general image, and are used by convolving them with the particular image at hand. The filter is usually only a few pixels wide, making the convolution efficient. Gaussian filters or derivatives of them (Marr, [1989]) are the most widely utilized. Among many other examples, filters utilizing basis functions have been used by Hueckel [1973] and Hummel [1979] in connection with edge detection (i.e. detecting *non*-smoothness), and by Haralick [1984] for derivatives. Recently, Meer and Weiss [1989] derived polynomial filters, based on a least square method, for smoothing and differentiation to high order. Another group of local filters is based on statistical "robust estimation" (Besl, [1988]). While they are very good at rejecting outliers, they are not always as good in dealing with the rest of the data, and for smoothing images with relatively little noise they are sometimes even worse than

the Gaussian. Besides, these filters are computationally intensive, requiring an iterative process.

While useful in many applications, many of the local filters are based on rather *ad hoc* assumptions, so it is not surprising that their output is sometimes too sensitive to both the input data and the particular details of the filter used. Often they contain internal parameters that need to be carefully fine tuned for each application in order to achieve a result that looks reasonable to humans. It is clear that the need exists for local filters based on a solid theoretical foundation such as the regularization theory discussed earlier.

Calculating derivatives has always been a desirable but very difficult goal in vision. Global regularization theory could be used for that purpose, but then the computational cost would increase even more. The available local filters have proven even more unreliable for derivatives than for smoothing. Our method applies differentiation by combining the reliability of the regularization method with the efficiency of the local filters.

We combine the two methods in the following way: we define a minimal curvature cost function and apply it to a basic "unit" image, i.e. an image with the value 1 at its center and 0 elsewhere. The continuous smoothing functional is translated to the discrete mesh using the "finite element" method, with bivariate Hermite splines as the basic interpolation functions. Minimizing the cost function, we obtain a smoothed version of the unit image. We use this minimal curvature version as a filter or mask to be convolved with any given image. We show that this intuitive filtering process can be justified mathematically by using a discrete equivalent of the Green function method used in solving differential equations. While most of the treatment involves the discrete case, we do study some properties of the continuous analog.

While these filters have merits in their own right, they can be of even greater usefulness within a larger context. The above treatment, like other treatments throughout the literature, assumes that the cost function has constant parameters. In a common physics analogy, the cost function is likened to the bending energy of a flexible plate that represents the surface. These methods fails when a surface has discontinuities, or sharp edges. A

usual remedy is to break the surface into smooth parts divided by the discontinuities, but this immediately raises the question of what is a discontinuity and how to determine it. Various *ad hoc* and not very satisfactory attempts have been made to solve the problem. Our further goal is to solve the problem by varying the parameters of the cost function in an *adaptive* way that will fit the local properties of the surface. A surface with rapid changes in its curvature needs a more flexible "plate" to describe it than does a slowly changing part of the image. By changing the physical parameters of the plate according to the local needs of the surface we can obtain good smoothing properties in every part of the surface without having to decide prematurely what is a discontinuity.

The problem was hard enough to solve with constant parameters because of the large system of linear equations that it involved. Non-constant parameters would make the problem nonlinear and almost beyond reach. However, our reduction of the solution to a simple convolution with filters not only makes the linear solution very easy, it also opens the way to a nonlinear solution. All we have to do now is to vary the parameters of the filter during the convolution in accordance with the local properties of the surface we deal with. (In a way, it is a "context-dependent" operation.) In a subsequent paper we will examine various ways of relating the filter parameters, or the flexibility of the smoothing plate, to the local average curvature of the surface.

In the next sections we calculate the cost function as a function of image variables by using 1-D basis functions, substitute particular spline basis functions, derive general solutions and put them in the form of a convolution.

2. The Cost Function

In this section we define the cost function to be minimized and express it in terms of unknown image variables. This is done by expressing the 2-D image by general basis function, whose coefficients are the unknowns.

The function to minimize is

$$E = \int\int_{\Omega} w(\mathbf{x})(f(\mathbf{x}) - g(\mathbf{x}))^2 d\mathbf{x} + \int\int_{\Omega} (\lambda^{xx} f_{xx}^2 + \lambda^{xy} f_{xy}^2 + \lambda^{yy} f_{yy}^2) d\mathbf{x} \qquad (1)$$

where f is the smoothed (unknown) image, g is the data and w are weights.

The first term is simply a measure of the distance between the data and the smoothed image, which is a quadratic norm with weights. The second part is the smoothing, or regularizing term. This is the term used in the Tikhonov and Arsenin theory to stabilize the ill posed problem, and can be understood by a physical analogy to a thin plate. The smoothing term, with its second derivatives, would represent a bending energy term of the plate, which we wish to minimize along with the distance of the plate from the data. The coefficients λ^α, with $\alpha = xx, xy, yy$, are the regularization coefficients which determine the amount of smoothing. The bigger the smoothing term, the smoother, but less conforming to the data, the output image will be. If $\lambda^{xx} = \lambda^{yy} = \lambda^{xy}/2$ one has the simple isotropic bending plate (Landau and Lifshitz, [1959]). In our subsequent work we will be interested in λ^α, w which are functions of both the coordinates and the direction.

We now have to discretize the problem. We could represent the derivatives in the smoothing term by finite differences, but our experience (Weiss, [1986]) and others' shows that the finite element method is more stable, besides producing a smooth shape even between the pixels. The idea is simply to represent the surface $f(\mathbf{x})$ as a linear combination of basis functions $\Phi_i(\mathbf{x})$:

$$f(\mathbf{x}) = \sum_i f_i \Phi_i(\mathbf{x}) \tag{2}$$

where f_i are unknown coefficients and $\Phi_i(\mathbf{x})$ are given basis functions. The cost function can now be written as

$$E = \int\int \sum_i \sum_\alpha \lambda^\alpha(\mathbf{x})(f_i \partial_\alpha \Phi_i)^2 d\mathbf{x} + \int\int \sum_i w(\mathbf{x})(f_i \Phi_i(\mathbf{x}) - g(\mathbf{x}))^2 d\mathbf{x}$$

Minimizing with respect to a particular coefficient f_j we obtain

$$\int\int \sum_i f_i \left\{ \sum_\alpha \lambda^\alpha (\partial_\alpha \Phi_i)(\partial_\alpha \Phi_j) + w \Phi_i \Phi_j \right\} d\mathbf{x} = \int\int w g \Phi_j d\mathbf{x}$$

This can be written in matrix form as

$$\sum_i K_{ij} f_i = G_j \tag{3}$$

where the vector G_j represents the image data

$$G_j = \int\int w(\mathbf{x})g(\mathbf{x})\Phi_j(\mathbf{x})d\mathbf{x} \tag{4}$$

and K_{ij} is the analog of a "stiffness matrix" of the plate:

$$K_{ij} = \int\int \left\{ w\Phi_i\Phi_j + \sum_\alpha \lambda^\alpha(\partial_\alpha\Phi_i)(\partial_\alpha\Phi_j) \right\} d\mathbf{x}$$

The first term above can be called the "mass" term K^{00}_{ij}, and the others are the bending terms, K^α. With obvious definitions one can now write

$$K_{ij} = K^{00} + K^{xx} + K^{xy} + K^{yy} \tag{5}$$

The goal of finding the unknowns f_i can now be accomplished by solving the linear system of equations (3) for the unknowns f_i. For a large system this is easier said than done and we will later describe an efficient direct method of solution. This method can be applied directly in the case of sparse data, in which the number of unknowns is relatively small.

3. Reduction to 1-D

So far we have dealt with general basis functions. Since all functions in 2-D can be expressed by basis functions that are products of 1-D functions, we will express K_{ij} in terms of such simpler functions. Thus we introduce the 1-D functions $\phi_k(x), \phi_k(y)$. For the decomposition one has to use different i, j indices for the x and y functions, i.e. i_x, i_y, j_x, j_y. The indices of the stiffness matrix i, j can be expressed as some convenient functions of these 1-D indices:

$$\begin{aligned} i &= i(i_x, i_y) \\ j &= j(j_x, j_y) \\ f_i &= f_{i_x i_y}, \qquad\qquad f_j = f_{j_x j_y} \\ \Phi_i(\mathbf{x}) &= \phi_{i_x}(x)\phi_{i_y}(y), \qquad \Phi_j(\mathbf{x}) = \phi_{j_x}(x)\phi_{j_y}(y) \end{aligned} \tag{7}$$

We will be able to express the above stiffness matrix terms in terms of three basic 1-D matrices, that need to be calculated only once for any given basis:

$$\begin{aligned} K^0_{i_x j_x} &= \int \phi_{i_x}(x)\phi_{j_x}(x)dx \\ K^1_{i_x j_x} &= \int \partial_x\phi_{i_x}(x)\partial_x\phi_{j_x}(x)dx \\ K^2_{i_x j_x} &= \int \partial_{xx}\phi_{i_x}(x)\partial_{xx}\phi_{j_x}(x)dx \end{aligned} \tag{8}$$

The matrices involving y are similar.

We will now express the terms K^{00}, K^{α} with $\alpha = xx, xy, yy$ in terms of the above basic matrices. The coefficients w, λ have been general so far, but to perform the decomposition we have to make simplifying assumptions about them. We assume that they are constant within the common support of Φ_i, Φ_j. We will later specialize to highly localized basis functions Φ_i, of the size of a pixel, so the approximation will be justified. These coefficients become functions of i, j rather than $\mathbf{x}$ so they can be put outside the integration.

We thus have for the mass term

$$\begin{aligned} K^{00}_{ij} &= \int\int w(\mathbf{x})\phi_{i_x}(x)\phi_{i_y}(y)\phi_{j_x}(x)\phi_{j_y}(y)dxdy \\ &= w_{ij}\int \phi_{i_x}(x)\phi_{j_x}(x)dx \int \phi_{i_y}(y)\phi_{j_y}(y)dy \\ &= w_{ij}K^0_{i_x j_x}K^0_{i_y, j_y} \end{aligned} \tag{9a}$$

For the bending terms we have, first for the x direction

$$\begin{aligned} K^{xx}_{ij} &= \int\int \lambda^{xx}(\mathbf{x})(\partial_{xx}\phi_{i_x}(x)\phi_{i_y}(y))(\partial_{xx}\phi_{j_x}(x)\phi_{j_y}(y))dxdy \\ &= \lambda^{xx}_{ij}\int \partial_{xx}\phi_{i_x}(x)\partial_{xx}\phi_{j_x}(x)dx \int \phi_{i_y}(y)\phi_{j_y}(y)dy \\ &= \lambda^{xx}_{ij}K^2_{i_x j_x}K^0_{i_y j_y} \end{aligned} \tag{9b}$$

and similarly for y derivative

$$K^{yy}_{ij} = \lambda^{yy}_{ij}K^0_{i_x j_x}K^2_{i_y j_y} \tag{9c}$$

For the mixed term

$$\begin{aligned} K_{ij}^{xy} &= \int\int \lambda^{xy}(\mathbf{x})(\partial_{xy}\phi_{i_x}(x)\phi_{i_y}(y))(\partial_{xy}\phi_{j_x}(x)\phi_{j_y}(y))dxdy \\ &= \lambda_{ij}^{xy}\int \partial_x\phi_{i_x}(x)\partial_x\phi_{j_x}(x)dx \int \partial_y\phi_{i_y}(y)\partial_y\phi_{j_y}(y)dy \\ &= \lambda_{ij}^{xy} K_{i_x j_x}^1 K_{i_y j_y}^1 \end{aligned} \tag{9d}$$

Substituting these expressions in the expression for the total stiffness matrix, eq.(5), completes the decomposition into 1-D quantities. In the next section we will specialize to specific basis functions.

4. The Basis Functions and the Stiffness Matrix

The general expressions for the stiffness matrix found in the previous section can be used with a variety of basis functions. Meer and Weiss [1989] have used various orthogonal polynomials for the special case of $\lambda^\alpha = 0$, i.e. no smoothing term. This case amounts to a simple least square fitting, and it was easily solved analytically because the stiffness matrix is diagonal. We obtained good smoothing properties but the derivatives were not as stable as we would like. Dierckx [1977] used cubic splines for the above special case, and obtained a system of equations that needs to be solved. In this paper we use spline functions with narrow local supports, i.e. finite elements, to make the solution more flexible locally. We then eliminate the need to solve a system of equations by deriving filters.

Because of the local support of the finite elements, the geometry of the image points plays an important role. There are many off the shelf programs for solving engineering problems with finite elements. In a general finite element code one has geometrical "nodes", i.e. points for which data is available or calculated. They are input manually so that a wide range of complicated geometries can be handled. We are interested here in a simple geometry that can be dealt with automatically, so we specialize to a square domain with a rectangular grid, the nodes being image pixels. Our cost function is somewhat different from what can be found in the ready made codes. For these reasons, after experimenting with some of these programs (ANSYS, ADINA) we were led to writing our own program.

The stiffness matrix elements found from (9) all involve products of two 2-D basis functions, each of which consists of two 1-D splines, and each node has several of these functions, so a good bookkeeping system becomes important.

Since the basis functions are nonzero only in neighborhoods of pixels, it s convenient to enumerate them in accordance with the enumeration of the pixels. The pixels are arranged on a rectangular grid and are denoted by m_x, m_y (going from 1 to maxima of M_x, M_y.) At each node (pixel) we have a small number of basis functions Φ_i, so the indices s_x, s_y are used to distinguish among basis functions belonging to the same pixel. These two indices enumerate the 1-D spline functions whose products make up the 2-D basis functions. Each basis function has two equivalent sets of indices: While the i (or j) indices enumerate all the functions in a linear list and are used for the rows and columns of the stiffness matrix, the m, s indices reflect the geometry of the problem.

A basis function can now be written as Φ_i, or as Φ_{m_x,m_y,s_x,s_y} or for brevity as $\Phi_{m,s}$. Similarly for its coefficient f_i. The 1-D functions can be written as ϕ_{m_x,s_x}, ϕ_{m_y,s_y}.

Fig. 1 shows a convenient way to relate to two kinds of indices for a 2×2 pixel example. More generally we can establish:

$$
\begin{aligned}
m_x &\in \{1, M_x\} && \text{node coordinate} \\
m_y &\in \{1, M_y\} && \text{node coordinate} \\
m &= (m_y - 1)M_x + m_x && \text{node number} \\
s_x, s_y &\in \{0, 1\} && \textit{intra node indices} \\
s &= 2s_x + s_y \\
i &= 4(m - 1) + s + 1 = 4(m_y - 1)M_x + 4m_x + 2s_x + s_y + 1 \\
j &= 4(m' - 1) + s' + 1 = 4(m'_y - 1)M_x + 4m'_x + 2s'_x + s'_y + 1
\end{aligned}
\tag{10}
$$

(We note that m', s' are not new indices, they are used to determine the column j of the stiffness matrix while m, s determine i.)

One can get a good grasp of the basic process described here without a detailed understanding of the plethora of indices used, but the intention here is to have enough detail for an actual implementation.

We now introduce the basis functions themselves. Taking the distance between pixels to be 1, we define two basic functions as zero outside the interval $[-1,1]$, while inside they are the two cubic Hermite spline polynomials (Fig 2):

$$\begin{aligned} \phi_{0,0}(x) &= (|x|-1)^2(2|x|+1) \\ \phi_{0,1}(x) &= x(|x|-1)^2 \end{aligned} \tag{11}$$

To form the whole basis these are shifted so they are centered around each pixel, in both the x and y directions, i.e.

$$\phi_{m_x,s_x}(x) = \phi_{0,s_x}(x-m_x) \tag{12}$$

$$f = \sum_i f_{i(m_x m_y s_x s_y)} \phi_{m_x s_x}(x) \phi_{m_y s_y}(y) \tag{13}$$

In this way we get functions with local support in a rectangular neighborhood around each pixel, extending only up to the next pixel. The Hermite polynomials are known to have good properties of convergence and reliability of interpolation up to second order derivatives (Strang and Fix, [1973]).

The first of the functions in eq. (11) has a value of 1 and a zero derivative at the center of the interval, while the second has a value of zero and a derivative 1. At the ends of the interval both the values and the derivatives vanish, so there is no (direct) influence on the value of the neighbors. Thus the coefficients f_i in the sum (13) are in fact values of the function f and its derivatives. It is easy to see that

$$\begin{aligned} f_{i(m_x m_y,0,0)} &= f(m_x,m_y) \\ f_{i(m_x m_y,1,0)} &= \partial_x f(m_x,m_y) \\ f_{i(m_x m_y,0,1)} &= \partial_y f(m_x,m_y) \\ f_{i(m_x m_y,1,1)} &= \partial_x \partial_y f(m_x,m_y) \end{aligned} \tag{14}$$

For higher derivatives, higher order splines are needed and more of them to a node, but the local character will not change.

Having taken care of the bookkeeping of the nodes and associated basis functions, we now have to perform the integrations of eq. (8). Each element of these matrices involve an integration over a pair of basis functions with a common support. Since the support is local, only a few of the pairs will have a common support. We can divide the image into rectangular tiles, with a node at each corner. Each tile supports only a small number of the basis functions. Thus one can save effort by going over all tiles and collecting the contribution of each to a matrix element of K^n, rather than going over all pairs i, j. This is known as the "assembly" process. (Fix and Strang, [1973], Bathe, [1982].) From the bookkeeping point of view, some indices are replaced by ones with a much more limited range:

$$m_x, m'_x \rightarrow m_x + \Delta m_x, \quad m_x + \Delta m'_x$$

$$m_y, m'_y \rightarrow m_y + \Delta m_y, \quad m_y + \Delta m'_y$$

where m_x, m_y are the coordinates of the bottom left node and $\Delta m_x, \Delta m_y$ are 0 or 1. The general indices i, j can be derived from them with eq. (10):

$$i = i(m_x + \Delta m_x,\ m_y + \Delta m_y, s_x, s_y)$$

$$j = j(m_x + \Delta m'_x,\ m_y + \Delta m'_y, s'_x, s'_y) \tag{15}$$

To assemble the matrices we now go over the indices $m_x, m_y, \Delta m_x, \Delta m_y, \Delta m'_x, \Delta m'_y, s, s'$ instead of $m_x, m_y, m'_x, m'_y, s, s'$.

Furthermore, since all tiles are similar it is enough to integrate over one tile and use the results in the others. The actual integration can be done in 1-D. We use the one-tile 1-D indices:

$$\hat{i}_x = 2\Delta m_x + s_x + 1$$

$$\hat{j}_x = 2\Delta m'_x + s'_x + 1$$

$$\hat{i}_y = 2\Delta m_y + s_y + 1$$

$$\hat{j}_y = 2\Delta m'_y + s'_y + 1 \tag{16}$$

which run from 1 to 4. The factor 2 above come from the fact that there are currently two basis functions per node.

Performing the integrations (8) on the one-tile 1-D case we obtain the 4×4 symmetric matrices, for an interval of length h

$$\hat{K}^0_{\hat{\imath}_x \hat{\jmath}_x} = \int_0^h \phi_{\hat{\imath}_x} \phi_{\hat{\jmath}_x} = \frac{h}{420} \begin{pmatrix} 156 & 22 & 54 & -13 \\ & 4 & 13 & -3 \\ & & 156 & -22 \\ & & & 4 \end{pmatrix}$$

$$\hat{K}^1_{\hat{\imath}_x \hat{\jmath}_x} = \int_0^h \partial_x \phi_{\hat{\imath}_x} \partial_x \phi_{\hat{\jmath}_x} = \frac{1}{30h} \begin{pmatrix} 36 & 3 & -36 & 3 \\ & 4 & -3 & -1 \\ & & 36 & -3 \\ & & & 4 \end{pmatrix}$$

$$\hat{K}^2_{\hat{\imath}_x \hat{\jmath}_x} = \int_0^h \partial_{xx} \phi_{\hat{\imath}_x} \partial_{xx} \phi_{\hat{\jmath}_x} = \frac{1}{h^3} \begin{pmatrix} 12 & 6 & -12 & 6 \\ & 4 & -6 & 2 \\ & & 12 & -6 \\ & & & 4 \end{pmatrix}$$

and similarly for y.

The above quantities are the contributions from each tile to the matrix elements of K^n $(n = 1, 2, 3)$:

$$\Delta K^n_{i_x j_x} = \hat{K}^n_{\hat{\imath}_x \hat{\jmath}_x} \tag{17}$$

and the addition to the total stiffness matrix is now given by eqs. (5,9) as

$$\Delta K_{i,j} = w_{m_x, m_y} \hat{K}^0_{\hat{\imath}_x \hat{\jmath}_x} \hat{K}^0_{\hat{\imath}_y \hat{\jmath}_y} + \lambda^{xx}_{m_x, m_y} \hat{K}^2_{\hat{\imath}_x \hat{\jmath}_x} \hat{K}^0_{\hat{\imath}_y \hat{\jmath}_y} + \lambda^{xy}_{m_x, m_y} \hat{K}^1_{\hat{\imath}_x \hat{\jmath}_x} \hat{K}^1_{\hat{\imath}_y \hat{\jmath}_y} + \lambda^{yy}_{m_x, m_y} \hat{K}^0_{\hat{\imath}_x \hat{\jmath}_x} \hat{K}^2_{\hat{\imath}_y \hat{\jmath}_y} \tag{18}$$

These terms are accumulated in the general stiffness matrix K_{ij}. There is usually a contribution from more than one tile to each element of K_{ij}.

We now summarize the process of assembling the stiffness matrix as follows:

For all image pixels m_x, m_y, for all $\Delta m_x, \Delta m_y, \Delta m'_x, \Delta m'_y$ equal 0 or 1, and for all s_x, s_y, s'_x, s'_y equal 0 or 1 do:

Calculate the one-tile indices $\hat{\imath}_x, \hat{\jmath}_x, \hat{\imath}_y, \hat{\jmath}_y$, eq. (16).

Retrieve the appropriate element of the 4×4 matrices $\hat{K}^0_{\hat{\imath}_x \hat{\jmath}_x}$,...etc.

Calculate the stiffness matrix indices i, j, eq. (10).

Calculate the contribution ΔK_{ij} from eq. (18) and add to the appropriate array element K_{ij}.

End do

This stiffness matrix contains all our assumptions about the geometry and smoothness of the problem. We are left with the algebraic problem of solving the system of equations that it represents, eq. (3). This is dealt with next.

5. Reducing the Solution to Small Filters

In theory, solving the system of equations (3) should be a straightforward application of algebraic methods. However, an image with 1024×1024, or 10^6 pixels generates a stiffness matrix with 10^{12} elements, and it would be a great waste of resources to deal with it in a straightforward way even if it could be done, which is doubtful. Fortunately, the matrix is very sparse, since only a few of the pairs of basis functions have a common support (i.e. the integrals (8) are non-zero). From eq. (10) one can derive the pairs i, j that have a non-zero element. Previously the system was solved using iterative methods (Terzopoulos [1988]). These methods are very efficient in use of memory, but many iterations over the whole image are needed to get close to convergence. Direct methods give a closed form solution but may need sophisticated hashing techniques to save memory, depending on the structure of the matrix. (Ortega, [1981].) In our case this structure is quite simple, being banded along the diagonal, so storage is quite simple. Direct methods are now used in most finite element codes (Bathe, [1982]) and we adopt one here.

In this section we show that the one can solve the problem by solving for small windows of the image, provided that we find a general solution for any small image. For this process

a direct method becomes even more attractive, as the small window makes it very easy to apply.

We begin by describing briefly the continuous analog of the method. We are given the differential equation, which we keep in 1-D for simplicity:

$$D_x f(x) = g(x)$$

where D_x is a differential operator, $f(x)$ is an unknown function and $g(x)$ is a given term, e.g. data (the inhomogeneous term). Now, instead of solving for each g one can solve for particularly simple data, i.e. delta functions, and build a general solution on this basis.

Thus we define the "Green function" $h(x,t)$ as the solution of

$$D_x h(x,t) = \delta(x-t)$$

The general solution is now given by

$$f = \int h(x,t) f(t) dx$$

Proof:

$$D_x f = \int D_x h(x,t) f(t) = \int \delta(x-t) f(t) = f(x)$$

If all the coefficients in the differential operator are constant, then the equation is shift invariant, i.e. we have the same Green function around each point t:

$$h(x,t) = h(x-t)$$

and thus the general solution above becomes a convolution.

We will later find the Green function for the continuous version of our smoothing problem.

We return now to our discrete case. We have (eq. (3)):

$$\sum_j K_{ij} f_j = G_i$$

The indices i, j are functions of the geometrical node numbers m, s (eq. (10)). We have

$$m = (m_x, m_y) \qquad s = s(s_x, s_y)$$

$$i = i(m, s) \qquad j = j(m', s')$$

and s enumerates the basis functions at each node. So we can write eq. (3) as

$$\sum_{m',s'} K_{m,m',s,s'} f_{m',s'} = G_{m,s}$$

We can simplify matters if we assume that the data is concentrated in the node, a standard practice. From eqs. (4,7,11) one can see that $G_{m,s}$ vanishes for all s except $s = 0$, in which case it is equal to the given data at the pixel m, $G_m \equiv G_{m,0}$. Thus the solution for the above equation vanishes for $s \neq 0$ and we can limit ourselves to the solution of

$$\sum_{m',s'} K_{m,m',0,s'} f_{m',s'} = G_m \tag{19}$$

In analogy with the continuous case we will now look for a solution to a system of equations with the data being a Kronecker delta rather than a general G:

$$\sum_{m',s'} K_{m,m',0,s'} h(m', s', n) = \delta_{mn} \tag{20}$$

where $n = n(n_x, n_y)$ is an arbitrary node of the grid.

The solution of (19) for general data is now

$$f_{m,s} = \sum_n h(m, s, n) G_n$$

Proof:

$$\sum_{m',s'} K_{m,m',0,s'} \sum_n h(m', s', n) G_n = \sum_n \delta_{mn} G_n = G_m$$

In dealing with images we observe that our problem is shift invariant in the node number m, because we constructed the finite element formalism in this way, i.e. the same basis functions just shifted unchanged from one pixel to another. (We assume for

the moment an unbounded image, analogous to the continuous case.) Thus our Green function h loses one variable and becomes a *filter* $h(m-n,s)$ which is *convolved* with the image:

$$f_{m,s} = \sum_n h(m-n,s)G_n \tag{21}$$

Furthermore, it is reasonable to assume that the quantities at pixel m will not be influenced very much by the data in a pixel n which is far away from m. Therefore the summation above does not need to be carried out over the whole image but can be limited to some rectangular window extending N_x, N_y pixels around m. Thus we can finally write the solution in term of the filters as:

$$f_{m_x m_y s_x s_y} = \sum_{\substack{n_x=-N_x,N_x \\ n_y=-N_y,N_y}} h(m_x-n_x, m_y-n_y, s_x, s_y)G_{n_x n_y} \tag{22}$$

Thus, the solution at the pixel m_x, m_y is given by a known mask applied to a window around that pixel.

One can see from (22) that there is a separate filter for each s_x, s_y, with eq. (14) giving the meaning of each one. Thus,

$$h(m_x, m_y, 0, 0)$$

will yield the smoothed function f itself, while

$$h(m_x, m_y, 1, 0)$$

is the x derivative, and the combinations (0,1), (1,1) yield the y and x,y derivatives respectively. Since convolution is associative, we can concatenate two first derivative filters to obtain one for the second derivative:

$$\partial_{xx} f(m) = h(m,1,0) \otimes (h(m,1,0) \otimes f(m)) = (h(m,1,0) \otimes (h(m,1,0)) \otimes f(m)$$

so that the filter for the second derivative is

$$h(m,2,0) = h(m,1,0) \otimes h(m,1,0)$$

and similarly for other derivatives. There is no point, however, in continuing the process beyond the order of the spline used: we need higher order splines than our cubics to obtain third and fourth derivatives.

In conclusion, we have reduced the regularization problem, which led to a large system of equations, to convolutions of the image with small filters. The filters can be derived once and for all (for a cost function with given factors) by solving eq. (20) for a small window, a relatively easy task. The results can be stored for use with any input images.

The next section summarizes the solution method used for the window.

6. Solving for the Filters

Deriving the filters amounts to solving the system of linear equations (20) for a small window. In principle, this could be done by the standard method of Gaussian elimination, but then one has to deal with the recurring problems of ill-conditioning and numerical instabilities. A method applicable to symmetric positive definite matrices is the Cholesky decomposition (Ortega, [1981]). It guarantees well conditioning and numerical stability. We describe here a modification for a general symmetric (non-singular) matrix. A stiffness matrix of a stable physical system is always positive definite. In our case it is at least symmetric.

We start with the decomposition of the symmetric matrix into a product:

$$K = LDL^T$$

where L is a lower triangular matrix having 1's on its diagonal

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 \\ L_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & 1 \end{pmatrix}$$

L^T is its transpose, an upper triangular matrix, and D is a diagonal matrix. Explicitly:

$$K_{ij} = \sum_{k=1}^{n} L_{ik} D_k L_{jk}$$

where n in the matrix size, in our case the total number of unknowns in the image. We further assume, without loss of generality, that the matrix is banded with p non-zero diagonals above and below the main diagonal. This is the case for our sparse stiffness matrix. The following iterative algorithm, based on the above summation, can be verified by induction:

For $j = 1, \ldots, n$ **do:**

$q \leftarrow \max(1, j - p)$

$D_k \leftarrow K_{jj} - \sum_{k=q}^{j-1} L_{jk}^2 D_k$

For $i = j+1, \ldots, \min(j+p, n)$ **do:**

$r \leftarrow \max(1, i - p)$

$L_{ij} \leftarrow (K_{ij} - \sum_{k=r}^{j-1} K_{ik} K_{jk} D_k)/D_j$

End do

End do

The output elements L_{ij} also form a banded matrix and can be conveniently stored by overwriting the corresponding input elements K_{ij} as soon they are calculated. The diagonal matrix elements D_k can be stored in place on the 1's in L. They also happen to be the eigenvalues of the matrix.

We can see from the limits of the iterations and the summation that the decomposition is of linear complexity in n and $O(p^2)$ in the bandwidth, i.e. it is $O(np^2)$. For a non-banded matrix we have $q = r = 1$ and the complexity would be n^3. Thus if the bandwidth is much smaller than the matrix dimension one obtains great savings.

For given data G the system of equations

$$Kf = G$$

can now be solved in two stages, with an intermediate solution vector f', as follows ("back substitution"):

$$\begin{aligned} Lf' &= G \\ DL^T f &= f' \end{aligned} \tag{24}$$

The first term can be written explicitly as

$$\sum_k L_{ik} f'_k = G_i$$

As before, by separating out the last term of the sum one obtains the iterative algorithm:

For $i = 1, \ldots, n$ **do:**

$$r \leftarrow \max(1, i - p)$$

$$f'_i \leftarrow G_i - \sum_{k=r}^{i-1} L_{ik} f'_k$$

End do

Similarly, the second stage is

$$D_i \sum_k L_{ki} f_k = f'_i$$

yielding the "mirror" algorithm

For $i = n, \ldots, 1$ **do:**

$$r \leftarrow \min(1, i + p)$$

$$f_i \leftarrow f'_i / D_i - \sum_{k=r}^{r+1} L_{ki} f_k$$

End do

For a banded matrix, the two stages are $O(np)$ in complexity, while for the non-banded case ($r = 1$) we have $O(n^2)$. Thus the main effort is in the decomposition stage, with $O(np^2)$. In our implementation the band width is proportional to the window side N and the matrix dimension to N^2 so we have a total of $O(N^4)$. A window with $N \approx 10$ takes a few seconds for a VAX to solve.

Finally, we have to substitute the appropriate data G in the above algorithms. Of course the algorithms could be used with the whole image itself as data. That would be useful for sparse data with a limited number of unknowns. In deriving the filters, however, the data is "unit" images (Kronecker δs) according to eq. (20). Denoting the central pixel in a window by n_x, n_y we want to solve the equation

$$K_{ij} f_{i(m_x, m_y, s)} = \delta_{m_x n_x} \delta_{m_y n_y}$$

The solution can be broken into S parts, one for each $s = 1, \ldots, S$, corresponding to our S filters (22). Algebraically, we have obtained one row of the inverse of the matrix and there is no need to invert the whole matrix.

7. The Continuous Approximation

For the continuous case the functional (1) can be minimized using the Euler-Lagrange equations of the calculus of variation, yielding the fourth order differential equation

$$\frac{\lambda}{w}\nabla^4\Phi(\mathbf{x}) + \Phi(\mathbf{x}) = g(\mathbf{x})$$

In cylindrical coordinates the rotationally symmetric Green function involves $N_0(\sqrt{i}r)$, i.e. Bessel functions of the second kind with complex variable. The function diverges at zero and its asymptotic behavior at infinity is unclear. In Cartesian coordinates we can learn the main features of the filters by examining the solution of the 1-D equation

$$4\sigma^4\frac{d^4\phi(x)}{dx^4} + \phi(x) = \delta(x-t)$$

where we have defined $4\sigma^4 \equiv \lambda/w$.

The Green function is

$$h(x-t) = \frac{1}{2\sigma}e^{-\frac{|x-t|}{2\sigma}}\cos\frac{x-t}{2\sigma}$$

Substituting $h(x-t)$ in the above equation, it is easy to verify that the left hand side vanishes for $x \neq t$, and that its integral has a jump of 1 over the discontinuity at $x = t$, (due to the jump in the third derivative), making it a delta function. The parameter σ was chosen so that the Green function will behave generally like a Gaussian so that the two can be compared. Fig. 3 shows the Green function in a solid line and the Gaussian. Fig. 4 shows their respective derivatives. These are used in convolution with an image to obtain an image derivative. We see that one major difference is in the central parts of the filters. The bending Green function is less smooth at the top, and its derivative is discontinuous. This can make a difference for features smaller than σ or high frequencies.

Indeed, the Fourier transform of the filters (Figs. 5,6) shows that the Gaussian filters, particularly the differentiation filter, attenuate higher frequencies more strongly than the bending filter. Another difference is the "bump" in the Fourier transform of the Green function, resulting from the $\cos(x/2\sigma)$ factor. Thus the function looks somewhat like a band-pass filter. A combination of such filters with different scales σ may be tailored to provide a flatter band-pass filter between given frequencies, something that can't be done with the Gaussian. (The expressions depicted are $e^{-\sigma^2/2}$ and $(2\sigma^2+1)/(4\sigma^4+1)$.)

8. Experiments

We solved the linear equations (20) for various window sizes and convolved the resulting filter with real images. Fig. 7 shows a cross section through the center of the smoothing filter, and Fig. 8 does the same with the filter for the first derivative. Fig. 9 shows a smoothed image, and Figs. 10,11,12 are derivatives $\partial_x, \partial_y, \partial_{yy}$ respectively. While any effects are hard to measure quantitatively for real images, we can see that the derivatives look much like one would expect. The horizontal edges almost disappear with the ∂_x filter and they show progressively more variation for increasing differentiation order $\partial_y, \partial_{yy}$. The smoother parts are progressively flattened.

Fig. 13 shows an isolated edge smoothed by the Green function (solid line) and a Gaussian. The Green smoothing correctly represents the top of the edge but needs some distance until it clings to the straight sides. This can be expected from a flexible plate that one tries to fit over the edge. The Gaussian smooths the edge away, consistent with a dissipating temperature gradient. Clearly both filters in their simple form would better be used on smoother parts of the image. A subsequent paper will deal with cost functions whose factors adapt locally to the properties of the image, so that σ will get smaller as the filter gets closer to the edge.

9. Conclusion

We have derived filters for smoothing and differentiation based on the principle of minimal curvature of surfaces, on rectangular grids. This principle is based on the regularization

theory that provides reliable stabilization of noisy or insufficient data. The derivation was numerical for the discrete case and analytic for the continuous approximation.

One can use certain parameters to control the degree of smoothing, unlike least square fitting, which is obtained as a special case. The filters turn out to have some general similarities to the Gaussian but they preserve some frequencies better than the Gaussian.

The filtered data can be used both as image values and derivatives at the pixels, and as factors that multiply spline functions interpolating between the pixels. The latter capability is particularly useful for sparse data. Other methods need to solve a large system of equations for these interpolation factors.

For irregular data with varying weights no simple filter method is applicable and a whole system of linear equations must be solved. The direct solution method we described is particularly stable and efficient for this purpose.

References

Barrow, H.G. and Tenenbaum, J.M. [1981], "Interpreting line drawings as three-dimensional surfaces", *Artif. Intell.*, **17**, 75–117.

Bathe, K.J. [1982], *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

Besl, P.J. [1988], "Robust window operators", Proceedings of the Second International Conference on Computer Vision, Tampa, FL.

Blake, A. and Zisserman, A. [1987], *Visual Reconstruction*, MIT Press, Cambridge, MA.

Dierckx, P. [1977], "An algorithm for least square fitting of cubic spline surfaces to functions on a rectilinear mesh over a rectangle", *J. Comp. & Appl. Math.*, **3**, 2.

Grimson, W.E.L. [1981], "The implicit constraint of the primal sketch", MIT A.I. Memo No. 663.

Hadamard, J. [1923], *Lectures on the Cauchy's Problem in Partial Differential Equations*, Yale Univ. Press, New Haven, CT.

Haralick, R.M. [1984], "Digital step edges from zero crossings of second directional derivatives", *IEEE Trans. PAMI-***6**, 58–68.

Horn, B.K.P. [1983], "The curve of least energy", *ACM Transactions on Mathematical Software*, **9**, 441–460.

Hueckel, M.F. [1973], "A local visual operator which recognizes edges and lines", *J. Assoc. Comp. Mach.*, **20**, 634–647.

Hummel, R.A. [1979], "Feature detection using basis functions", *Computer Graphics & Image Processing*, **9**, 40–55.

Landau, L.D. and Lifshitz, E.M. [1959], *Theory of Elasticity*, Pergamon Press, Riverside, NJ.

Marr, D. [1982], *Vision*, Freeman, San Francisco, CA.

Meer, P. and Weiss, I. [1989], "Smoothed differentiation filters for images", Univ. of Maryland, Center for Automation Research TR 424.

Ortega, J.M. and Poole, W.G. [1981], *An Introduction to Numerical Analysis for Differential Equations*, Pitman, Marshfield, MA.

Poggio, T. [1987], Proceedings of the Image Understanding Workshop, Los Angeles, CA.

Strang, G. and Fix, G.J. [1973], *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ.

Terzopoulos, D. [1988], "The computation of visible surface representation", *IEEE Trans. PAMI-***10**, 417–438.

Weiss, I. [1986], "Curve fitting with optimal mesh point placement", University of Maryland, Center for Automation Research TR 22.

Weiss, I. [1988], "3-D shape representation by contours", *Computer Vision, Graphics & Image Processing*, **41**, 80–100.

Figure Captions

Fig. 1. Enumeration schemes for local basis functions

Fig. 2. Cubic Hermite splines

Fig. 3. Smoothing filters: minimal-curvature (solid line) and Gaussian

Fig. 4. Differentiation filters: minimal-curvature (solid line) and Gaussian

Fig. 5. Fourier transform of the smoothing filters

Fig. 6. Fourier transform of the differentiation filters

Fig. 7. Discrete minimal-curvature filter for smoothing

Fig. 8. Discrete minimal-curvature filter for differentiation

Fig. 9. Smoothed image

Fig. 10. x-derivative of image

Fig. 11. y-derivative of image

Fig. 12. Second y-derivative of image

Fig. 13. Edge smoothing

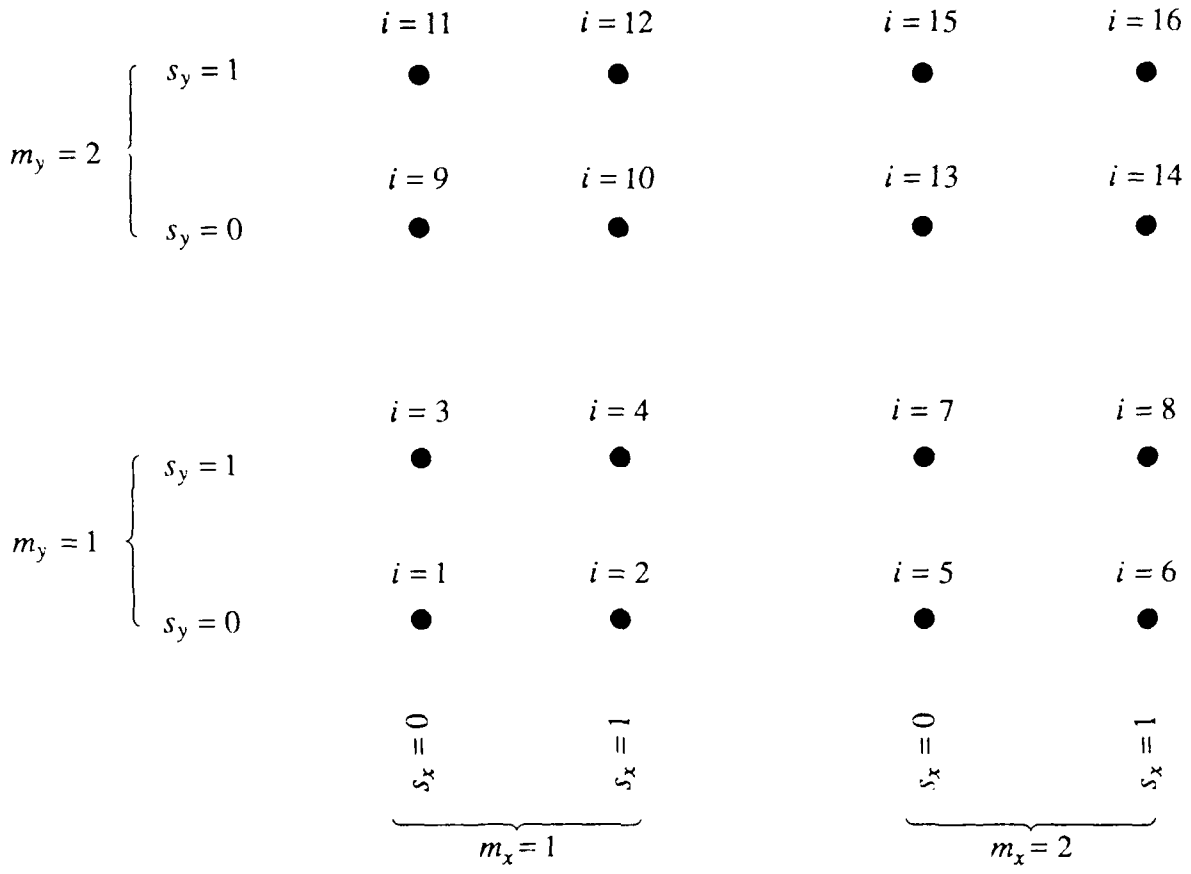

i = 11
i = 12
i = 15
i = 16
$m_y = 2$
$s_y = 1$
i = 9
i = 10
i = 13
i = 14
$s_y = 0$
i = 3
i = 4
i = 7
i = 8
$m_y = 1$
$s_y = 1$
i = 1
i = 2
i = 5
i = 6
$s_y = 0$
$s_x = 0$
$s_x = 1$
$s_x = 0$
$s_x = 1$
$m_x = 1$
$m_x = 2$

Figure 1

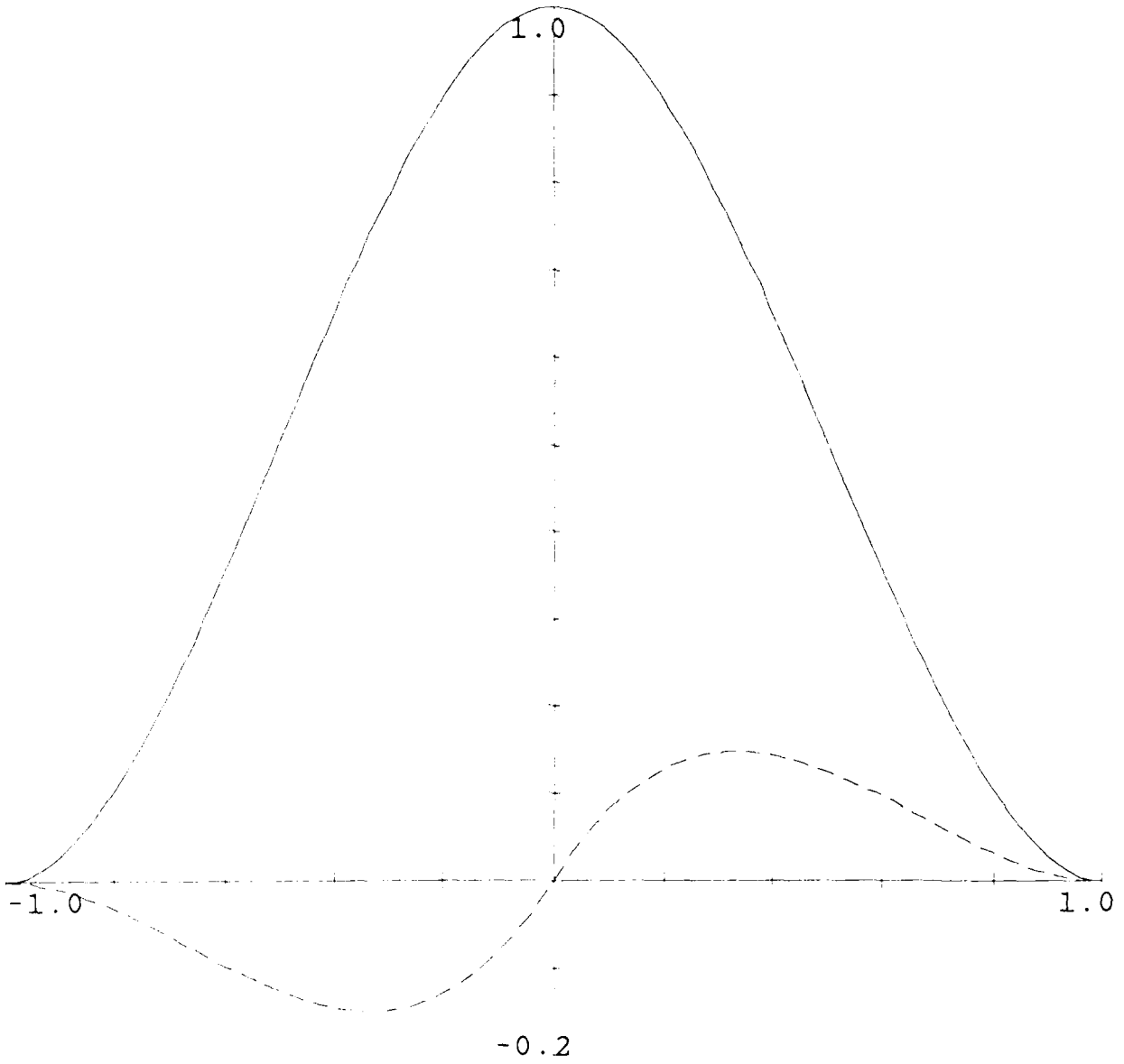
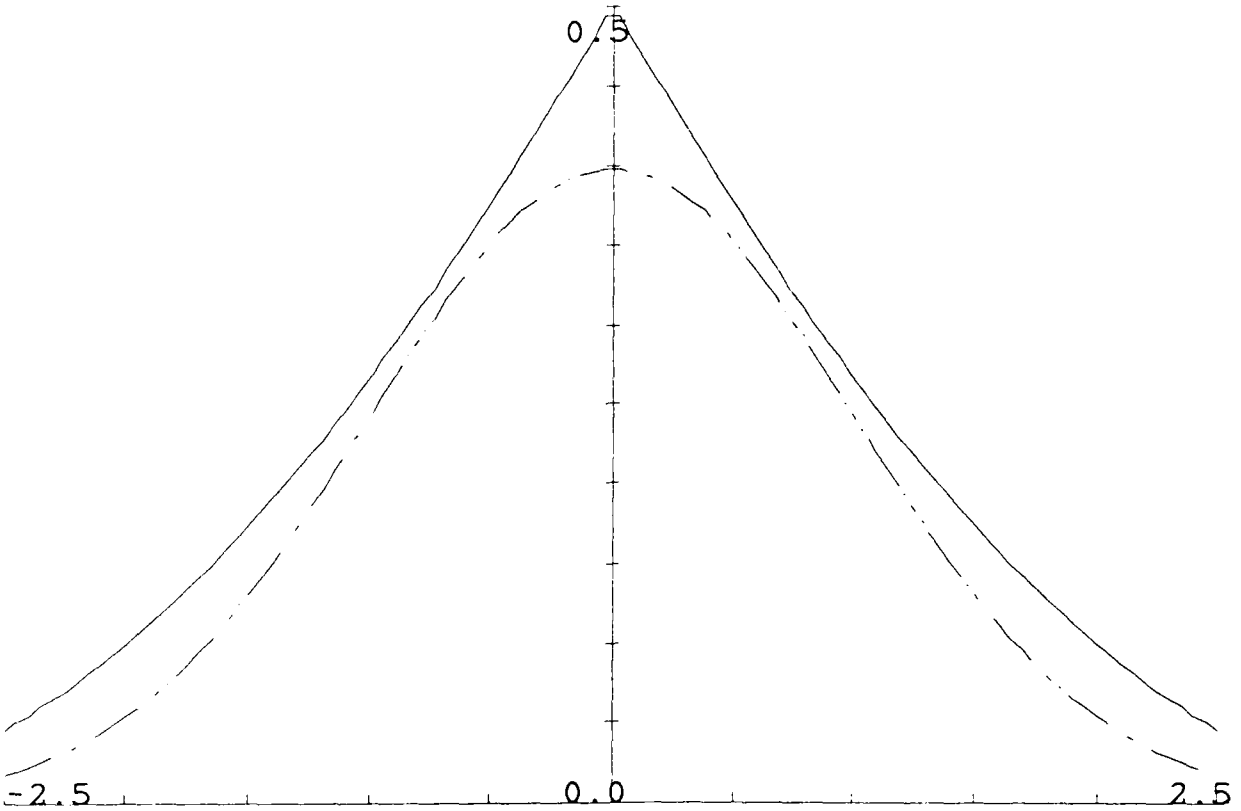
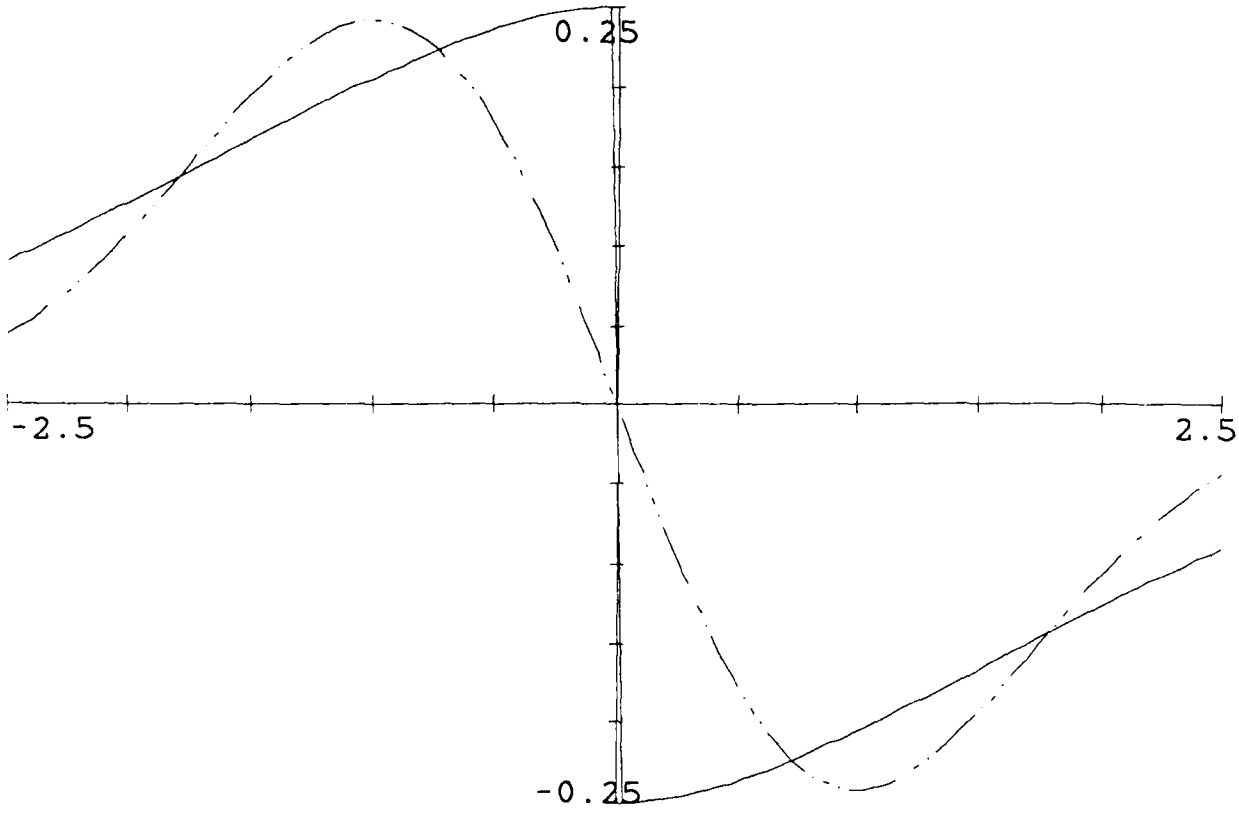


Figure 2



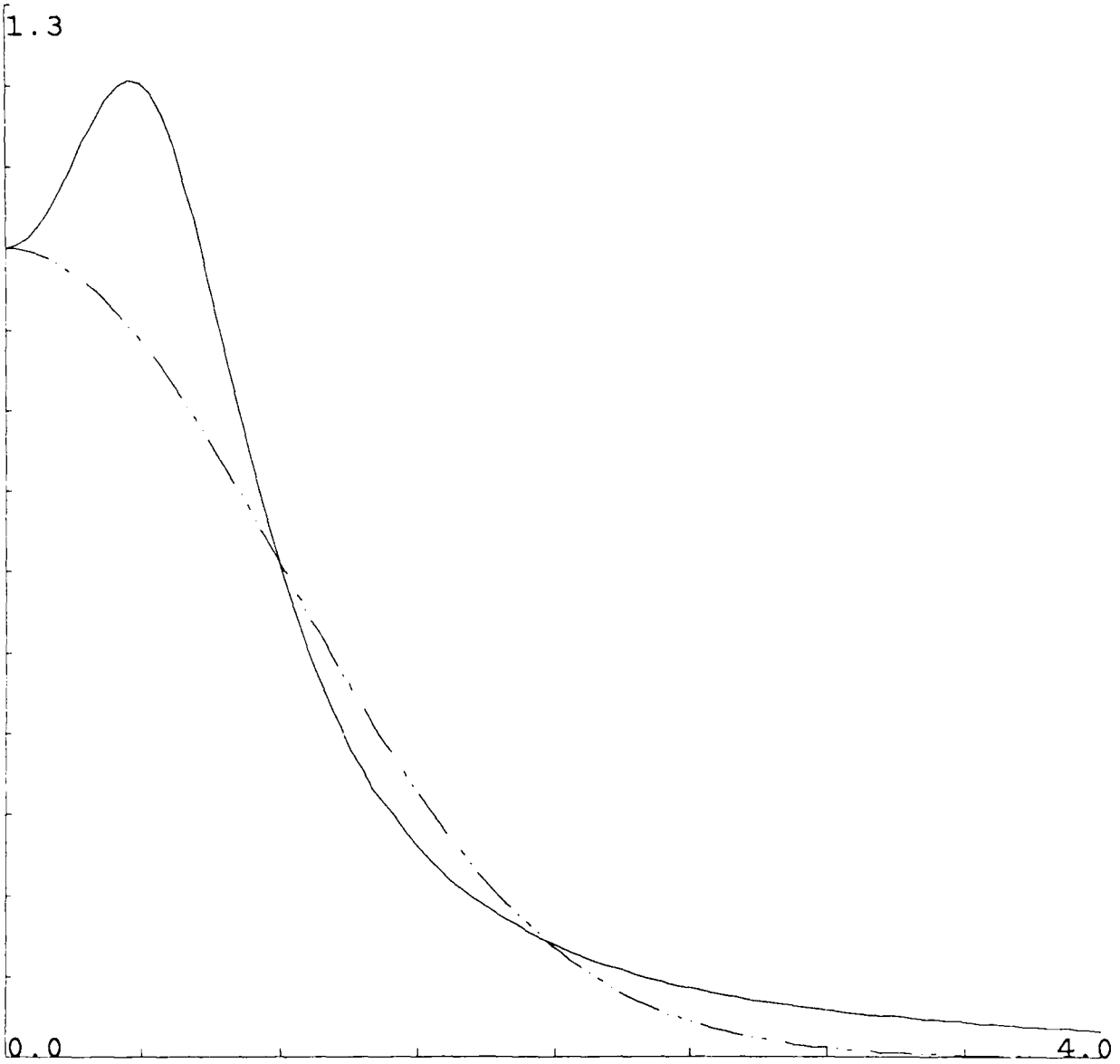
SMOOTHING FILTERS: MINIMAL-CURVATURE AND GAUSSIAN

Figure 3



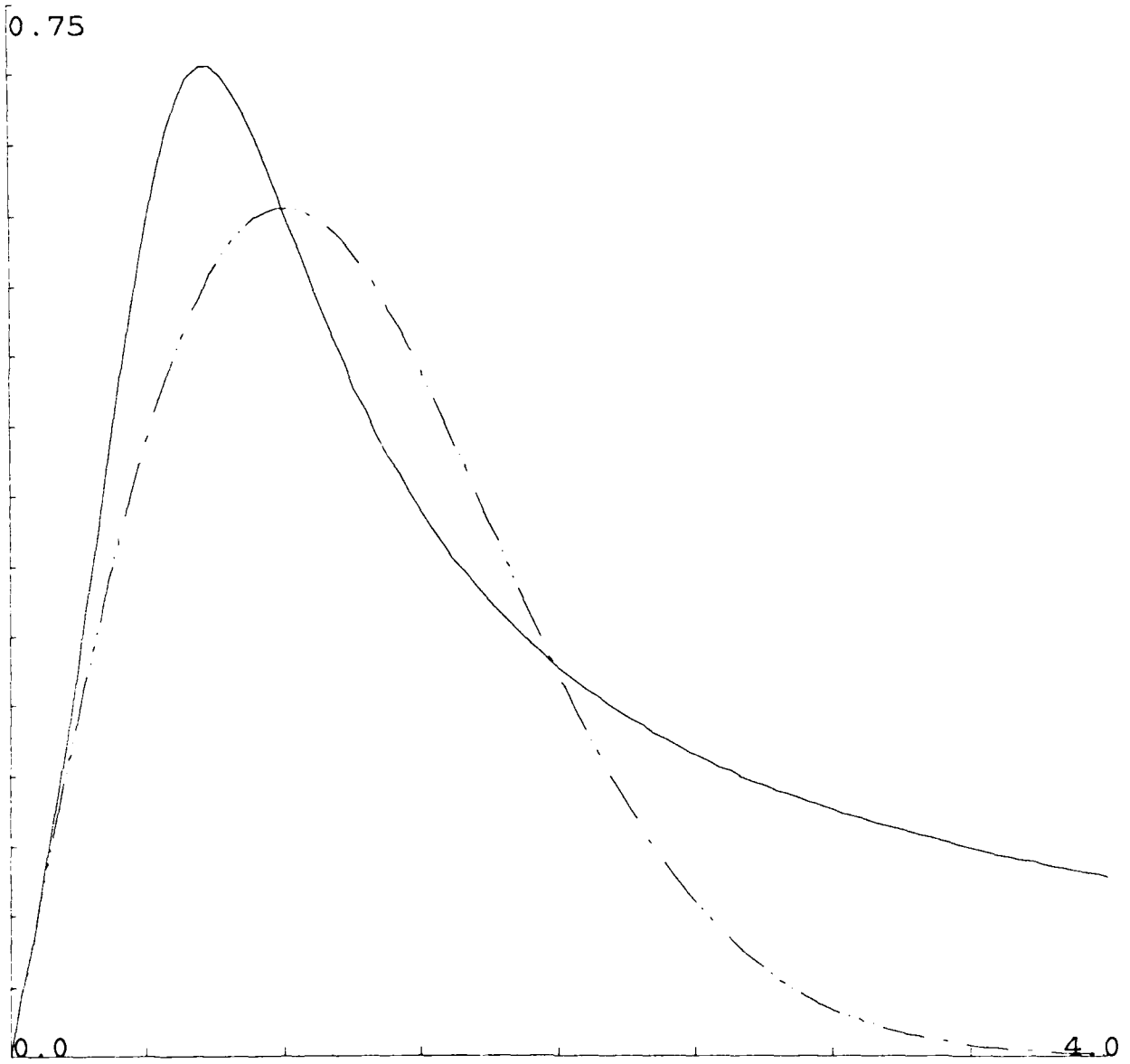
DIFFERENTIATION FILTERS: MINIMAL-CURVATURE AND GAUSSIAN

Figure 4



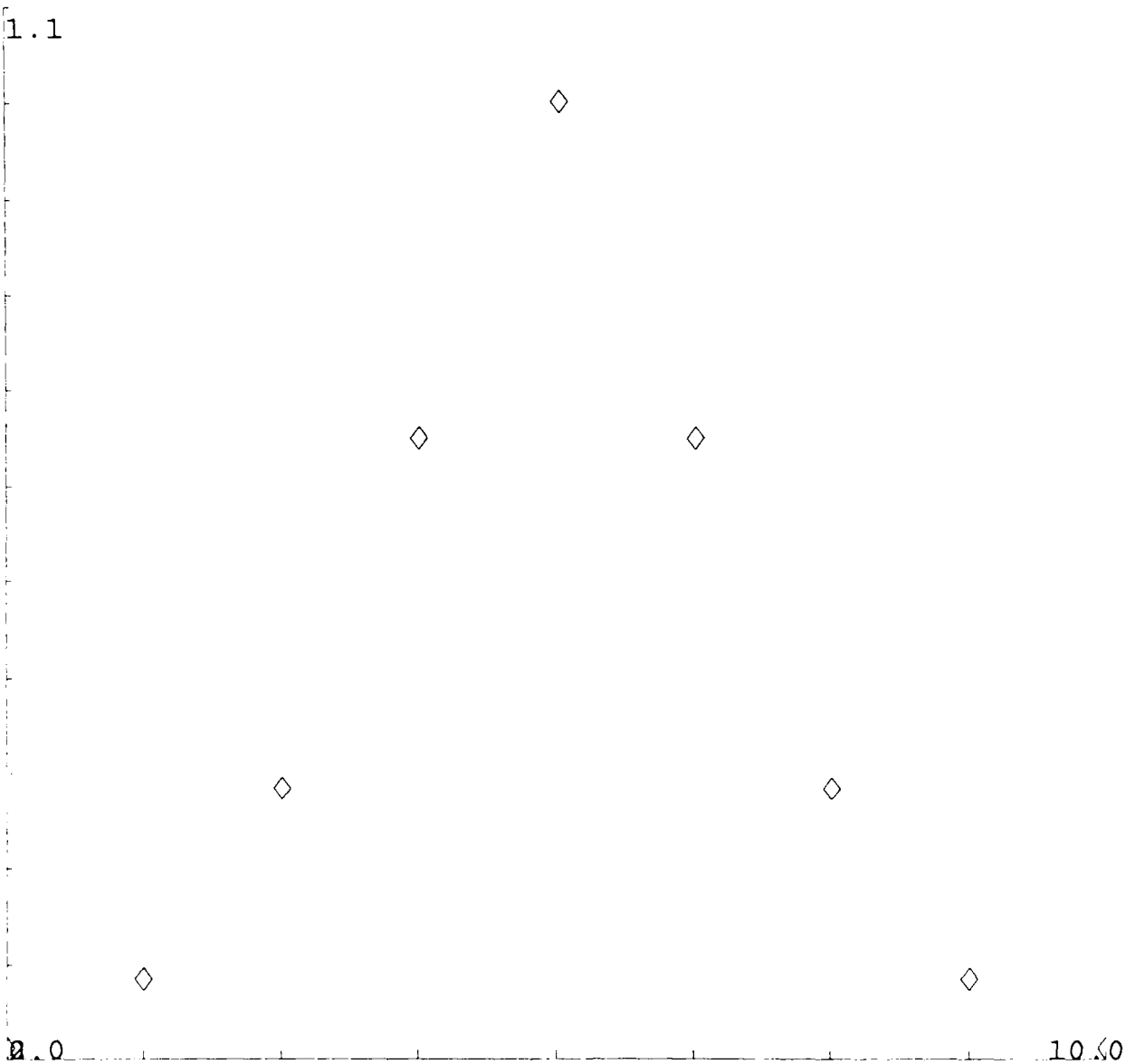
FOURIER TRANSFORM OF SMOOTHING FILTERS

Figure 5



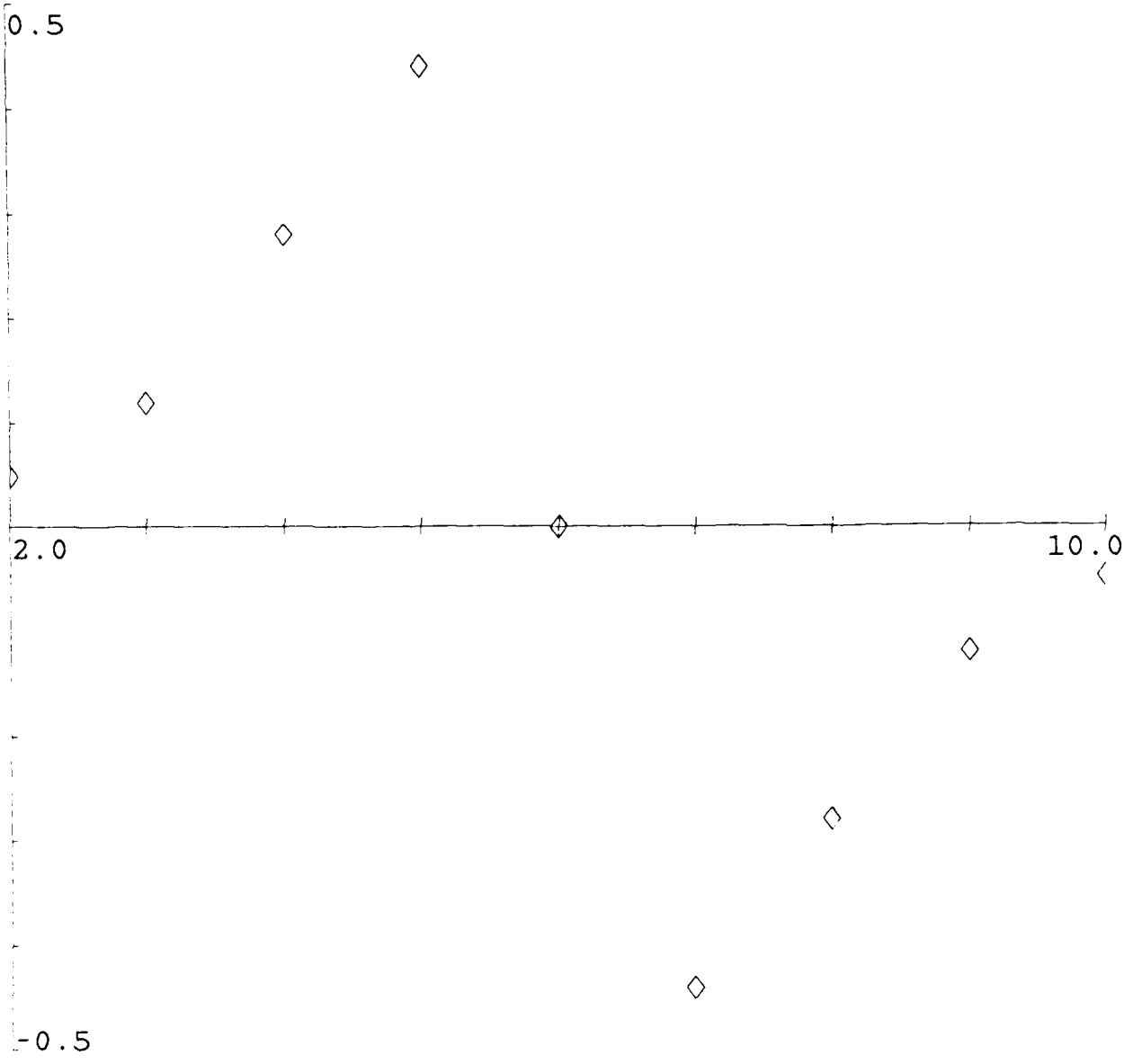
FOURIER TRANSFORM OF DIFFERENTIATION FILTERS

Figure 6



DISCRETE MINIMAL-CURVATURE FILTER FOR SMOOTHING

Figure 7



DISCRETE MINIMAL-CURVATURE FILTER FOR DIFFERENTIATION

Figure 8

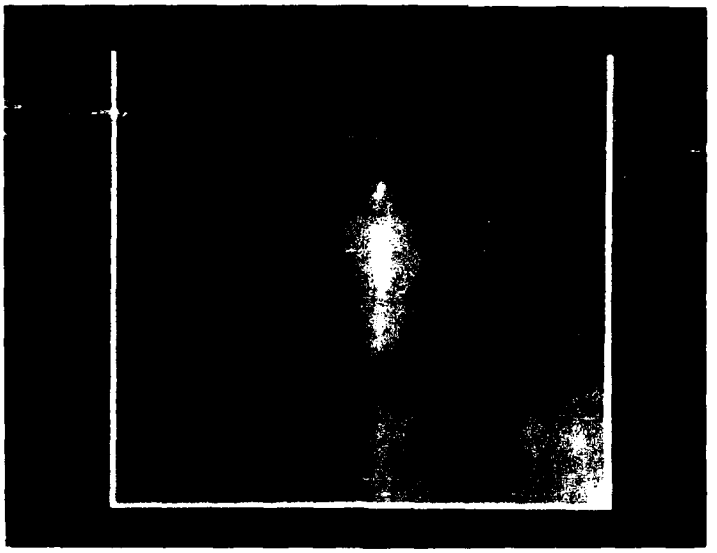


Figure 9

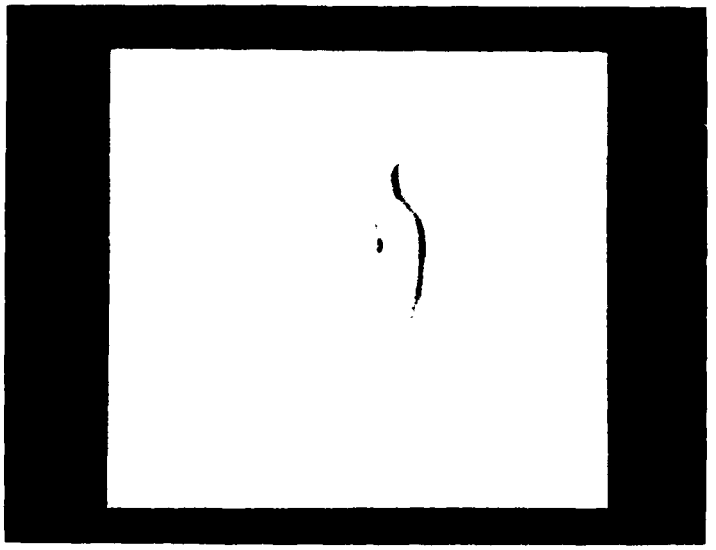


Figure 10

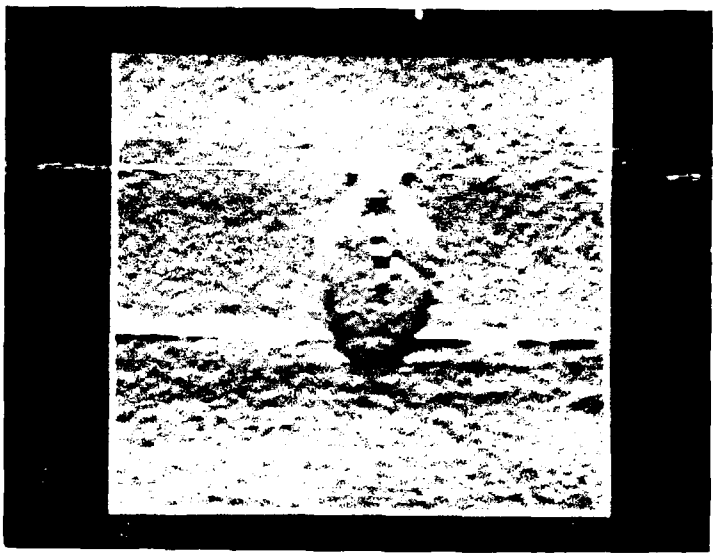


Figure 11

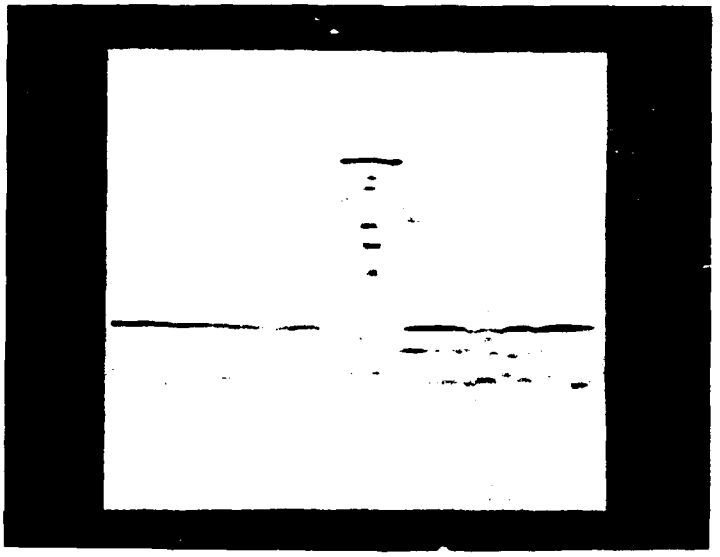
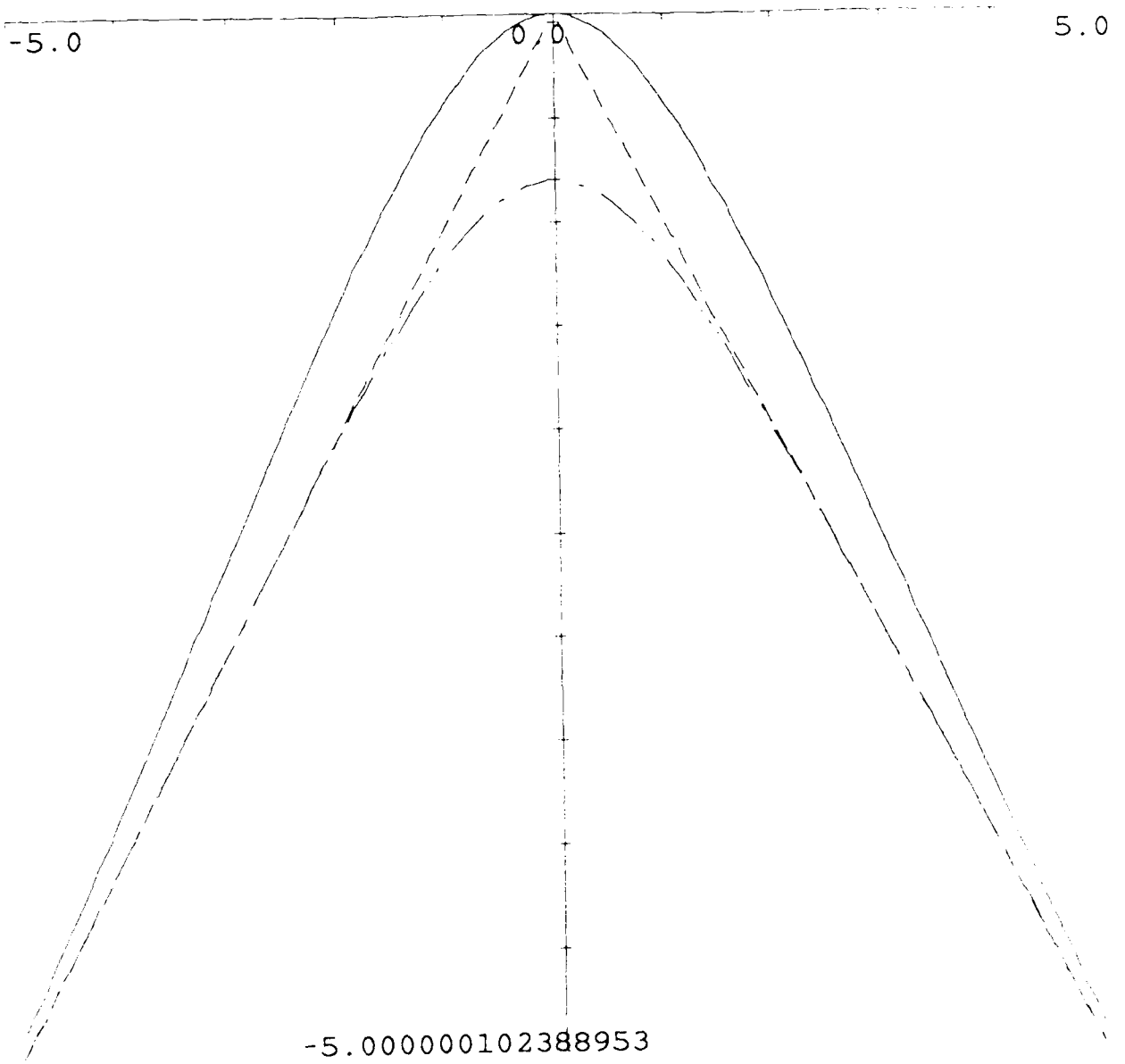


Figure 12



EDGE SMOOTHING

Figure 13

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS N/A
2a SECURITY CLASSIFICATION AUTHORITY N/A	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b DECLASSIFICATION DOWNGRADING SCHEDULE N/A	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) CAR-TR-470 CS-TR-2341	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION University of Maryland 6b OFFICE SYMBOL (If applicable) N/A	7a NAME OF MONITORING ORGANIZATION Department of the Navy Office of the Chief of Naval Research
6c ADDRESS (City, State, and ZIP Code) Center for Automation Research College Park, MD 20742-3411	7b ADDRESS (City, State, and ZIP Code) 800 North Quincy Street Arlington, VA 22217-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION Dept. of the Navy Ofc. Chief of Naval Research 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-89-J-1854
8c ADDRESS (City, State, and ZIP Code) 800 North Quincy Street, Code 1513:BSM Arlington, VA 22217-5000	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

IMAGE SMOOTHING AND DIFFERENTIATION WITH MINIMAL-CURVATURE FILTERS

12 PERSONAL AUTHOR(S)
Isaac Weiss

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM ____ TO N/A	November 1989	38

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The fundamental problem of smoothing and differentiating of noisy images has been previously approached in two different ways: 1) Minimization of a smoothness functional, a theoretically well understood procedure but one that involves the solution of a very large system of equations involving all the pixels of the image, for each image. 2) Use of small scale, ready made filters for local smoothing. The process is computationally cheap but generally ad hoc and not very realiable. This paper offers a way to combine the advantages of the two approaches. We construct general filters for local windows of the image, derived from maximization of smoothness or "regularization" theory. In this way the theoretically robust minimization process becomes suitable for practical implementation, possibly in real time, and is readily adaptable to local image properties. Filters for more reliable derivatives are also being derived.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c OFFICE SYMBOL